A New Spectral Method for Latent Variable Models

Matteo Ruffini, Marta Casanellas, and Ricard Gavaldà

Universitat Politècnica de Catalunya

November 11, 2017

Abstract

This paper presents an algorithm for the unsupervised learning of latent variable models based on the method of moments. We present efficient estimates of the moments for two models that are well known e.g. in text mining, the single topic model and Latent Dirichlet Allocation, and we provide a tensor decomposition algorithm for the moments that proves to be robust both in theory and in practice. Experiments on synthetic data show that the proposed estimators outperform the existing ones in terms of reconstruction accuracy, and that the proposed tensor decomposition technique achieves the learning accuracy of the state-of-the-art method with significantly smaller running times. We also provide examples of applications to real-world text corpora for both single topic model and LDA, obtaining meaningful results.

1 Introduction

Latent variable models (LVM) are a wide class of parametric models characterized by the presence of some hidden *unobservable* variables influencing observable data. A lot of widely used models belong to this class: Gaussian Mixtures, Latent Dirichlet Allocation, Naïve Bayes and Hidden Markov Models and many others. The huge availability of data and the widespread of real world applications, like health-care data mining, text analytics, vision and speech processing, has boosted the need for accurate and fast algorithms to learn models belonging to this class.

For many decades, the standard approach to learn such models has been the Expectation Maximization method (EM) Dempster et al. (1977), an iterative technique based on the maximum likelihood principle. EM is easy to understand and to implement and can be used for any latent variable model. Nevertheless, EM has many drawbacks: it is known to produce suboptimal results and might be very slow when the model or data dimension grows (Balle et al., 2014).

To overcome these issues a variety of techniques exploiting Method of Moments and tensor decomposition have been proposed in the last few years Anandkumar et al. (2014, 2012a,b); Jain and Oh (2014); Hsu and Kakade (2013); Balle et al. (2014); Chaganty and Liang (2013); Song et al. (2011). Generalizing an approach introduced in (Pearson, 1894), these methods first store in multidimensional tensors the low order moments of the observable data, and then recover the

E-mail: matteo.ruffini@estudiant.upc.edu, marta.casanellas@upc.edu, gavalda@cs.upc.edu

parameters of the desired model via tensor decomposition techniques. Moment-based algorithms are in general faster than EM, as they only require a single pass through the data and have provable guarantees of learning accuracy in polynomial time.

In Anandkumar et al. (2014), the authors presented an exhaustive survey showing that the spectral learning of most of the known LVM could be abstracted in two steps: first, given a prescribed LVM, they show how to operate with the data to obtain an empirical estimate of the moments, expressed in the form of a symmetric low rank three-dimensional tensor; in a second step, this tensor is decomposed to obtain the unknown parameters of the model. That paper also provides a method to decompose the retrieved tensor and obtain the unknown model parameters: the *Tensor Power Method* (TPM). Such abstraction allows to split the research on Method of Moments algorithms in two main non-disjoint areas.

1. A first area concentrates on methods to retrieve from data empirical estimates of the moments for specific LVMs. Examples can be found in (Jain and Oh, 2014) for mixture models with categorical observations, (Chaganty and Liang, 2013) for mixtures of linear regressions, in (Anandkumar et al., 2012a) for Naïve Bayes Models or in Hsu et al. (2012) for Hidden Markov Models. Important examples come from text mining, where the observable data, called *features*, generally coincides with the words appearing in a document, while the hidden variable can be, for example, the topic of the document. A simple model is the *single topic model*, where each text is assumed to be about a unique topic and the probability of a given word of appearing in a text depends on the topic of the text; equations to retrieve a moment representation for this model are provided in (Anandkumar et al., 2012a) and in (Zou et al., 2013). An alternative and more complex model is Latent Dirichlet Allocation (LDA) (see Griffiths and Steyvers, 2004; Blei et al., 2003), for which empirical estimators of the moments have been provided in (Anandkumar et al., 2012b).

2. A second area focuses on methods to decompose the symmetric tensors of the moments obtained from any LVM, and retrieve the parameters of the model. The reference method here is the *Tensor Power Method* (TPM) from Anandkumar et al. (2014), a method that decomposes the input moments using a three dimensional extension of the well-known matrix power method. The main issue of this algorithm is its scalability, as its computational complexity depends on a factor k^5 where k is the number of latent components. A viable alternative, that in general has better dependence on the number of latent factors, consists in using matrix-based techniques and a *simultaneous diagonalization* approach. Examples of these methods can be found in Anandkumar et al. (2012a), where an algorithm based on the eigenvectors of a linear operator is used, and in Anandkumar et al. (2012b) with a method based on the singular vectors of a singular-value decomposition (SVD). These two methods are far faster than TPM, but they are sensitive to perturbations on the input data. A different line of work consists in approaching the tensor decomposition problem with matrix optimization methods to perform simultaneous diagonalization (Kuleshov et al., 2015) or simultaneous Schur decomposition (Colombo and Vlassis, 2016). These techniques provide appealing experimental results, but provide global optimality guarantees only in limited settings and suffer of poor computational performance if compared with the methods listed above.

A method of moments uses a set of N samples to empirically estimate the moments and then uses tensor decomposition methods to recover the model parameters from the decompositions of the

estimated, perturbed, moments. The decomposition methods listed above differ from each other on how perturbations of the input moments affect the final results. Method of moments can thus be improved acting in two distinct points; first acting on the procedures providing empirical estimates of the moments, by reducing their dependence on the sample size, so to get more accurate estimates with less data. Second, by working on the decomposition algorithms, reducing their sensitiveness to input perturbations and improving their performance. This paper aims to contribute on both these areas; in particular:

- We present improved estimators of the moments for the single topic model and Latent Dirichlet Allocation. These estimators modify the ones presented in (Zou et al., 2013) increasing the robustness and the stability with respect to the noise, as experimentally shown in Section 5. Also, we present a novel theorem that relates the sample accuracy of the proposed estimates to the sample size and to the lengths of the documents.
- We provide a new algorithm (named SVTD, ***S**ingular **V**alue based **T**ensor **D**ecomposition*) to decompose the retrieved low-rank symmetric tensors of the moments. This method is alternative to the ones presented in the cited literature, and is based on the singular values of a SVD, which are known to be stable under random perturbations (unlike the singular vectors, as shown in Stewart 1990). Our algorithm is simple to implement and to understand, is deterministic and based only on standard matrix operations. Experimental results (see Section 5) show that SVTD outperforms existing matrix-based methods from Anandkumar et al. (2012a,b) in terms of reconstruction accuracy, and is an order of magnitude faster than TPM. In Remark 3.4 we outline in more detail the differences between the presented method and the existing techniques.
- Finally, we test SVTD on real world text corpora, both for single topic model and LDA, with satisfactory and meaningful results.

The outline of the paper is the following: Section 2 contains the proposed estimators of the moments and a sample complexity bound; Section 3 contains the proposed decomposition algorithm; Section 4 contains a perturbation analysis; Section 5 tests the presented methods on both synthetic and real-world data; Section 6 concludes the paper outlining possible future developments and applications.

2 Moments estimation for Latent Variable Models

2.1 An improved estimator for the single topic model

In this section we recall the single topic model for which we introduce a new estimator of the moments; we then provide a sample accuracy bound for this estimator and we compare it with existing methods.

We consider a corpus of N text documents and a set of k topics; each document is deemed to belong to only one topic. The vocabulary appearing in the corpus is constituted of n words, from which it is immediate to label all the words of the vocabulary with a number between 1 and n. The generative process works as follows:

- First, a (hidden) topic $Y \in \{1, ..., k\}$ is drawn, according to a given probability distribution; we define, for any $j \in \{1, ..., k\}$ the probability of drawing the topic j as follows:

$$\omega_j := \mathbb{P}(Y = j), \text{ and } \omega := (\omega_1, \dots, \omega_k)^\top.$$

- Once the topic has been chosen, all the words of the documents are generated according to a multinomial distribution; for each $i \in \{1, ..., n\}$, $\mu_{i,j}$ will be the probability of generating word i under topic j:

$$\mathbb{P}(\text{Drawing word } i | Y = j) = \mu_{i,j}, \text{ and } M = (\mu_{i,j})_{i,j} \in \mathbb{R}^{n \times k}.$$

  Also we will denote with μ_i the set of columns of M: $M = [\mu_1|, ..., |\mu_k]$. It is a common practice to identify a topic with the probability distribution of the words under that topic, i.e. with the columns μ_1,...,μ_k of M.

A practical encoding of the words in a document consists in identifying each word with an $n-$tuple $x \in \mathbb{R}^n$, defined as:

$$(x)_h := \begin{cases} 1 & \text{the word is } h, \\ 0 & \text{else.} \end{cases}$$

In fact, if x_j is the $j-$th word of a document of c words, we can define X as a vector whose coordinate h represents the number of times the word h has appeared in the document: $X := \sum_{j=1}^{c} x_j$. It is common to call X a *bag of words* representation of a document. We can see that, if the topic is j, each coordinate of X is distributed as a binomial distribution with parameter c and $\mu_{i,j}$:

$$Distr(X_i | Y = j) \approx B(c, \mu_{i,j})$$

We now assume to have a corpus of N documents; for each document $i \in \{1, ..., N\}$ we assume to have the word-count vector $X^{(i)}$, and the total number of words in the document: $c_i = \sum_{j=1}^{n} (X^{(i)})_j$. These are the only variables that we assume known, while all the parameters of the model, i.e. the pair (M, ω), and the hidden topic of each document are supposed to be unknown.

Remark 2.1. Recovering the model parameters is a useful step to infer the hidden topic of each document in a corpus. In fact, given a set of parameters (M, ω), and a document X, if Y is the hidden topic of X we can calculate

$$\mathbb{P}(Y = j | X) = \frac{\mathbb{P}(X | Y = j)\, \omega_j}{\sum_{i=1}^{k} \mathbb{P}(X | Y = i)\, \omega_i}$$

and assign X to the topic that maximizes that probability.

The following theorem is a variation of Propositions 3 and 4 in (Zou et al., 2013) and relates the observable moments of the known variables with the unknowns (M, ω). It will provide three estimators: $\tilde{M}_1 \in \mathbb{R}^n$, $\tilde{M}_2 \in \mathbb{R}^{n \times n}$ and $\tilde{M}_3 \in \mathbb{R}^{n \times n \times n}$ converging to the symmetric low rank tensors that will be used to retrieve the model parameters.

Theorem 2.1. Fix a value $N \in \mathbb{N}$, and let $X^{(1)}, ..., X^{(N)}$ be N sample documents generated according to a single topic model with parameters (M, ω). Define the vector $\tilde{M}_1 \in \mathbb{R}^n$, and the

The proofs of all the results are provided in the appendix.

symmetric tensors $\tilde{M}_2 \in \mathbb{R}^{n\times n}$ and $\tilde{M}_3 \in \mathbb{R}^{n\times n\times n}$ whose entries are as follows, for $h \leq l \leq m$:

$$
\begin{aligned}
(\tilde{M}_1)_h &= \frac{\sum_{i=1}^N (X^{(i)})_h}{\sum_{i=1}^N c_i}, \\
(\tilde{M}_2)_{h,l} &= \frac{\sum_{i=1}^N (X^{(i)})_h (X^{(i)} - 1\chi_{h=l})_l}{\sum_{i=1}^N (c_i - 1)c_i}, \\
(\tilde{M}_3)_{h,l,m} &= \chi_{h<l<m} \frac{\sum_{i=1}^N (X^{(i)})_h (X^{(i)})_l (X^{(i)})_m}{\sum_{i=1}^N (c_i - 2)(c_i - 1)c_i}. \\
&+ \chi_{h=l<m \vee h<l=m} \frac{\sum_{i=1}^N (X^{(i)})_h (X^{(i)} - 1)_l (X^{(i)})_m}{\sum_{i=1}^N (c_i - 2)(c_i - 1)c_i}. \\
&+ \chi_{h=l=m} \frac{\sum_{i=1}^N (X^{(i)})_h (X^{(i)} - 1)_l (X^{(i)} - 2)_m}{\sum_{i=1}^N (c_i - 2)(c_i - 1)c_i}.
\end{aligned}
$$

where χ is the indicator function and c_i is the number of words appearing in document i. We have:

$$
\mathbb{E}[\tilde{M}_1] = M_1 := \sum_{i=1}^k \omega_i \mu_i, \qquad \mathbb{E}[\tilde{M}_2] = M_2 := \sum_{i=1}^k \omega_i \mu_i \otimes \mu_i, \qquad \mathbb{E}[\tilde{M}_3] = M_3 := \sum_{i=1}^k \omega_i \mu_i \otimes \mu_i \otimes \mu_i.
$$

where $\otimes$ denotes the tensor product between vectors.

Notice that, because M_2 ans M_3 are symmetric, the previous theorem defines all the entries of that operators. Given a sample, we are able to calculate the three estimators $\tilde{M}_1$, $\tilde{M}_2$ and $\tilde{M}_3$. Theorem 2.1 allows to express observable moments in the form of a symmetric tensor. By construction it is immediate to see that both M_2 and M_3 have symmetric-rank less than or equal to k, and that $\lim_{N\to\infty} \tilde{M}_i = M_i$ for $i = 1, 2, 3$.

The following theorem provides a sample accuracy bound for the estimators of Theorem 2.1.

Theorem 2.2. Let $\tilde{M}_2$ and $\tilde{M}_3$ the empirical estimates of M_2 and M_3 obtained using Theorem 2.1; define also

$$
C_1 = \sum_{i=1}^N c_i, \quad C_2 = \sum_{i=1}^N (c_i - 1)c_i, \quad C_3 = \sum_{i=1}^N (c_i - 2)(c_i - 1)c_i
$$

$$
W_2^{(N)} = \frac{\sum_{i=1}^N (c_i(c_i - 1))^2}{C_2^2}, \quad W_3^{(N)} = \frac{\sum_{i=1}^N (c_i(c_i - 1)(c_i - 2))^2}{C_3^2}
$$

then, for any $0 \leq \delta < 1$, we have that

$$
\mathbb{P}(||M_2 - \tilde{M}_2||_F < \epsilon) > 1 - \delta
$$

holds for any corpus whose document lengths $(c_1, ..., c_N)$ satisfy

$$
\sqrt{W_2^{(N)}(1 - ||M_2||_F^2)} + \sqrt{\log(\frac{1}{\delta})} \frac{\max_j(c_j)\sqrt{C_1}}{C_2} < \epsilon.
$$

Also, for any pair $\epsilon > 0$ and $\delta > 0$, we have that

$$P(||M_3 - \tilde{M}_3||_F < \epsilon) > 1 - \delta$$

holds when

$$\sqrt{W_3^{(N)}(1 - ||M_3||_F^2)} + \sqrt{\log(\frac{1}{\delta})}\frac{\max_j(c_j(c_j-1))\sqrt{C_1}}{C_3} < \epsilon$$

Remark 2.2. We briefly comment on the results of the theorem. We focus on M_2 (similar arguments hold for M_3), analyzing the case where all the documents have the same length c (so, for all i, $c_i = c$) and c is somewhat large (so $c \simeq c - 1$). Then the bound simplifies to:

$$\sqrt{\frac{1}{N}(1 - ||M_2||_F^2)} + \sqrt{\frac{1}{Nc}\log(\frac{1}{\delta})}.$$

It is interesting to notice that the worst-case accuracy of the bound is $\epsilon = O(1/\sqrt{N})$. Also, the bound becomes smaller as c is large, with a clear limitation: if we have very few documents (N small), even if they are very long (large c), it is impossible to accurately learn the model, as in particular we may not even see all the topics.

Remark 2.3 (Alternative estimates of the moments)**.** The most simple technique to obtain from a text corpus described as in this section a symmetric low-rank tensor expression is the one described in (Anandkumar et al., 2012a), that, for each document $i \in \{1, ..., N\}$ considers three randomly selected words, $x_1^{(i)}, x_2^{(i)}, x_3^{(i)}$, and then shows that

$$\frac{\sum_{i=1}^N (x_1^{(i)})_h}{N} \xrightarrow[N\to\infty]{} (M_1)_h, \quad \frac{\sum_{i=1}^N (x_1^{(i)})_h (x_2^{(i)})_l}{N} \xrightarrow[N\to\infty]{} (M_2)_{h,l},$$

$$\frac{\sum_{i=1}^N (x_1^{(i)})_h (x_2^{(i)})_l (x_3^{(i)})_m}{N} \xrightarrow[N\to\infty]{} (M_3)_{h,l,m}.$$

This method only uses three words per documents, and has mainly illustrative purposes, as noticed by the authors. A method more similar to the one proposed in Theorem 2.1 is described in (Zou et al., 2013). Both, the method in (Zou et al., 2013) and Theorem 2.1, average the estimators with the document lengths, taking into consideration all the words in each document; however in (Zou et al., 2013), the averaging is done for each document and then they are averaged together with the same weight; for example, the off-diagonal entries of $\tilde{M}_2$, in (Zou et al., 2013) are calculated as follows:

$$(\tilde{M}_2)_{h,l} := \frac{1}{N}\sum_{i=1}^N \frac{(X^{(i)})_h (X^{(i)})_l}{(c_i - 1)c_i}$$

Such calculation is a simple average of many estimators, that gives to all the documents the same weight: $\frac{1}{N}$. Instead, in Theorem 2.1, we propose the following different formula:

$$(\tilde{M}_2)_{h,l} := \frac{\sum_{i=1}^N (X^{(i)})_h (X^{(i)})_l}{\sum_i (c_i - 1)c_i} = \sum_{i=1}^N \frac{(X^{(i)})_h (X^{(i)})_l}{(c_i - 1)c_i}\frac{(c_i - 1)c_i}{\sum_j (c_j - 1)c_j}$$

We can see that here we perform a weighted average, where the weight of the sample i is $\frac{(c_i-1)c_i}{\sum_j (c_j-1)c_j}$, giving in practice more weight to longer documents, which are supposed to be the most reliable; we will experimentally see in Section 5 that the proposed approach is less sensitive to the noise, providing improved results. If all the documents have the same length, the two estimates will produce the same number.

2.2 Extension to Latent Dirichlet Allocation

In this section we extend the results of the previous section to a more complex latent variable model, the Latent Dirichlet Allocation.

The obvious criticism of the single topic model is that each document can deal with a unique topic, an hypothesis that is commonly considered unrealistic. To overcome this issue, more complex models have been introduced, and one of these is Latent Dirichlet Allocation (LDA) (Griffiths and Steyvers, 2004; Blei et al., 2003). In its simplest form, LDA assumes that each document deals with a multitude of topics, in proportions that are governed by the outcome of a Dirichlet distribution. More precisely, considering our text corpus with N documents with a vocabulary of n words, the generative process for each text is the following:

- First a vector of topic proportions is drawn from a Dirichlet distribution with parameter $\alpha \in \mathbb{R}^k_+$, $Dir(\alpha)$; we recall that Dirichlet distribution is distributed over the symplex

$$\Delta^{k-1} = \{v \in \mathbb{R}^k : \forall i, v_i \in [0,1],\ and\ \sum v_i = 1\}$$

  and has the following density function, for $h \in \Delta^{k-1}$:

$$\mathbb{P}(h) = \frac{\Gamma(\alpha_0)\prod_{i=1}^k h_i^{\alpha_i - 1}}{\prod_{i=1}^k \Gamma(\alpha_i)}$$

  Where $\alpha_0 = \sum \alpha_i$. From a practical point of view, this step consists in drawing a vector of parameters $h \in \Delta^{k-1}$ such that h_i represents the proportion of the topic i in the document.

- Once the topic proportions (also named *mixture of topics*) have been designed, each word of the document is generated according to the following procedure: first a (hidden) topic of the word, say $Y \in \{1, ..., k\}$, is drawn, according to the probabilities defined by h (so we will have probability h_j of drawing topic j) and then we will generate the word itself according to a multinomial distribution; for each $i \in \{1, ..., n\}$, $\mu_{i,j}$ will be the probability of generating the word i under the topic j:

$$\mathbb{P}(\text{Drawing word } i | Y = j) = \mu_{i,j},\ \ and\ \ M = (\mu_{i,j})_{i,j} \in \mathbb{R}^{n \times k}.$$

Maintaining the notation of the previous section, we will indicate $x_j^{(i)} \in \mathbb{R}^n$ as the coordinate vector indicating the word at position j in document i, $X^{(i)} = \sum x_i$ as the word-count vector of document i and $c_i = \sum_{j=1}^n (X^{(i)})_j$ as the number of words in that document. In the case of LDA, the unknown model parameters are M and α, so the pair (M, α).

As in the case of the single topic model, we want to manipulate the observable moments in order to obtain a set of symmetric low rank tensors. The following theorem is an immediate modification of the one presented in (Anandkumar et al., 2012b, Lemma 3.2), and relates the observable moments of the known variables with the unknowns (M, α); providing the required representation. The only modification consists in the fact that we have used the estimates of Theorem 2.1 instead of the standard ones of Remark 2.3.

Theorem 2.3. *Let $\tilde{M}_1, \tilde{M}_2$ and $\tilde{M}_3$ the empirical estimates defined in Theorem 2.1. Define*

$$\tilde{M}_2^\alpha := \tilde{M}_2 - \frac{\alpha_0}{\alpha_0+1}\tilde{M}_1 \otimes \tilde{M}_1$$

$$\tilde{M}_3^\alpha := \tilde{M}_3 - \frac{\alpha_0}{\alpha_0+2}(M_{1,2}) + \frac{2\alpha_0^2}{(\alpha_0+2)(\alpha_0+1)}\tilde{M}_1 \otimes \tilde{M}_1 \otimes \tilde{M}_1$$

where $M_{1,2} \in \mathbb{R}^{n\times n\times n}$ is a three dimensional tensor such that

$$(M_{1,2})_{h,l,m} = ((\tilde{M}_2)_{h,l}(\tilde{M}_1)_m + (\tilde{M}_2)_{l,m}(\tilde{M}_1)_h + (\tilde{M}_2)_{m,h}(\tilde{M}_1)_l)$$

Then

$$\mathbb{E}[\tilde{M}_2^\alpha] = \sum_{i=1}^{k} \frac{\alpha_i}{(\alpha_0+1)\alpha_0}\mu_i \otimes \mu_i = M_2^\alpha$$

$$\mathbb{E}[\tilde{M}_3^\alpha] = \sum_{i=1}^{k} \frac{2\alpha_i}{(\alpha_0+2)(\alpha_0+1)\alpha_0}\mu_i \otimes \mu_i \otimes \mu_i = M_3^\alpha$$

This technique allows to express observable moments in the form of a symmetric tensor. Both M_2^α and M_3^α have symmetric-rank less than or equal to k, and so we can use any tensor decomposition algorithm to retrieve the unknown model parameters (M, α) from them. A major advantage of this theorem, as of the homologous theorem in (Anandkumar et al., 2012b), is that it only requires the knowledge of the value α_0, while non-spectral methods require the knowledge of the full vector α.

Remark 2.4 (Inference)**.** Similarly to the single topic model, one of the main usages of LDA is to infer the mixture of hidden topics of each document in a corpus. Unfortunately, an exact formula to perform this inference is not known, but a number of approximate approaches exist, like Gibbs sampling (Griffiths and Steyvers, 2004; Newman et al., 2009) and Expectation Propagation (Blei et al., 2003). In our case, if we assume to know the values of model parameters (M, α), we can apply a modified Gibbs Sampling to infer the topic mixture for a given text; consider a text, whose words are $x_1, ..., x_c$; then, in LDA, each word x_i is generated by a unique topic Y_{x_i}. Using the equations for Gibbs Sampling from Griffiths and Steyvers (2004), if Y_{x_i} is the hidden topic of word x_i and Y_{-x_i} is the set of topic assignment for all the words in the document excluded x_i, it can be shown that

$$\mathbb{P}(Y_{x_i} = j | Y_{-x_i}, x_i) \approx \mu_{x_i,j}\frac{n_{-i,j} + \alpha_i}{c - 1 + \alpha_0} \tag{1}$$

where $n_{-i,j}$ is the number of words assigned to topic j excluding x_i, c is the total number of words in the document and $\mu_{x_i,j}$ is the probability of drawing the word x_i under topic j. So, given a document, first we have to assign to each word a hidden topic, and then update this assignment word by word in a iterative way, using a monte-carlo assignment governed by equation (1). Each iteration updates the number of words assigned to a given topic; after a suitable number of iterations, we can estimate the topic mixture for a given document as the vector $h \in \mathbb{R}^k$ such that

$$(h)_j = \frac{n_j + \alpha_i}{c + \alpha_0}$$

where n_j is the number of words assigned to topic j.

2.3 Moments representation for other Latent Variable Models

In the previous sections we have seen, for the single topic model and for LDA, how to estimate from data three entities M_1, M_2 and M_3 of the form

$$M_1 = \sum_{i=1}^{k} \omega_i \mu_i, \tag{2}$$

$$M_2 = \sum_{i=1}^{k} \omega_i \mu_i \otimes \mu_i, \tag{3}$$

$$M_3 = \sum_{i=1}^{k} \omega_i \mu_i \otimes \mu_i \otimes \mu_i. \tag{4}$$

Other, less simple, models admit a similar characterizations, allowing to take a dataset and obtain, via linear transformations an empirical estimate of the triple (M_1, M_2, M_3). Examples are Gaussian Mixture Models Hsu and Kakade (2013); Ge et al. (2015), Hidden Markov Models Balle et al. (2014) latent trees Huang et al. (2014) and mixtures of linear regressions Chaganty and Liang (2013). This characterization reduces the problem of learning the parameters of a LVM to a tensor/matrix decomposition problem. In particular, the task becomes to provide an algorithm $\mathcal{A}$ able to map the triple (M_1, M_2, M_3) into the pair of model parameters (M, ω). To this task is dedicated the next section.

3 Singular Value-based Tensor Decomposition

In this section we present an algorithm to retrieve the parameters of a LVM from the triple (M_1, M_2, M_3). The proposed method is alternative to the ones presented in the cited literature, only relies on matrix decomposition techniques and it is deterministic and scalable. Experimental comparisons (see Section 5) show that this method provides state of the art accuracy performances with an improved scalability.

Our method relies on the observation that M_2 and the slices of M_3 admit a representation in term of matrix products as

$$M_2 = M \Omega M^\top \tag{5}$$

$$M_{3,r} = M \Omega^{1/2} diag(m_r) \Omega^{1/2} M^\top \tag{6}$$

where $\Omega = diag(\omega)$, $M_{3,r} \in \mathbb{R}^{n \times n}$ is the rth slice of M_3 and m_r is the r-th row of M. As a first step, it stores the whitened slices of M_3 into a three dimensional tensor H in $\mathbb{R}^{n \times k \times k}$ and then performs n SVD on the slices of H (belonging to $\mathbb{R}^{k \times k}$) obtaining the rows of M as the singular values of that slices; for this reason we name our method SVTD, ***S**ingular **V**alue based **T**ensor **D**ecomposition*. The key steps of our algorithm (SVTD) are outlined in Algorithm (1).

The constructive proof of the following theorem will explain why SVTD performs a correct retrieval of the desired model parameters.

Theorem 3.1. *Let r be the feature selected at Step 3 of SVTD. If all the elements of m_r are distinct and M_2 and M_3 have rank k, then SVTD produces exactly the values of (M, ω).*

Algorithm 1 SVTD

Require: M_1, M_2, M_3, and the number of latent states k

1: Decompose M_2 as $M_2 = U_k S_k U_k^\top$ with a SVD truncated at the k-th singular vector.
2: Define the whitening matrix $E = U_k S_k^{1/2}$ and calculate its pseudoinverse $E^\dagger = (S_k)^{-1/2} U_k^\top$.
3: Select a feature r and compute $M_{3,r}$
4: Compute O as the singular vectors of $H_r := E^\dagger M_{3,r} {E^\dagger}^\top$
and m_r as the singular values.
5: **for** $i = 1 \to n$ **do**
6: Compute $H_i := E^\dagger M_{3,i} {E^\dagger}^\top$
7: Obtain the $i-$th row of M as the diagonal entries of $O^\top H_i O$
8: **end for**
9: Obtain ω solving $M_1 = M\omega$
10: Return (M, ω)

Proof. As a first step we perform a SVD $M_2 = U_k S_k U_k^\top$, where $U_k \in \mathbb{R}^{n\times k}$ and $S_k \in \mathbb{R}^{k\times k}$ are the matrices of the singular vectors and values truncated at the $k - th$ greatest singular value. Then we define the whitening matrix $E = U_k S_k^{1/2} \in \mathbb{R}^{n\times k}$, and for each slice $M_{3,r}$ of M_3, we define $H_r \in \mathbb{R}^{k\times k}$ as

$$H_r = E^\dagger M_{3,r} (E^\top)^\dagger,$$

where $E^\dagger = (S_k)^{-1/2} U_k^\top$ is the Moore-Penrose pseudo-inverse of E. Observing that there exists a unique orthonormal $k \times k$ matrix O, such that

$$M\Omega^{1/2} = EO, \tag{7}$$

one gets the following characterization of $M_{3,r}$:

$$M_{3,r} = M\Omega^{1/2} diag(m_r) \Omega^{1/2} M^\top = EO diag(m_r) O^\top E^\top,$$

from which it follows that

$$H_r = O diag(m_r) O^\top.$$

Now, one gets the r-th row of M as the *singular values* of H_r. Repeating these steps for all the $i \in \{1, ..., n\}$ will provide the full matrix M. In order to avoid ordering issues with the columns of the retrieved M, one can use the same matrix O to diagonalize all the matrices H_i, as O is uniquely determined, because the elements of m_r are distinct. So, compute O as the singular vectors of H_r, for a certain feature r, and re-use it for all the other features. The subsequent estimations of ω is straightforward, and can be obtained by solving the linear system $M_1 = M\omega$. □

Remark 3.1 (On the selection of feature r)**.** The initial steps of the algorithm need the isolation of a feature r to compute the matrix O. While theoretically we could select any feature r such that all elements of $m_r = (\mu_{r,1}, ..., \mu_{r,k})$ are distinct, it is clear that, if matrices M_1, M_2 and M_3 are subject to perturbations, the results obtained by the algorithm might vary a lot, depending on the selected feature r. Theorem 4.1 will show that the accuracy of the algorithm under perturbed data will depend on how different are the elements in $m_r = (\mu_{r,1}, ..., \mu_{r,k})$. A consequence of this is that a good way to find feature r, and so a reliable matrix O, is to repeat the steps 3 and 4 of the algorithm, isolating different features and selecting the one that maximizes the quantity $\min_{i\neq j}(|\mu_{r,i} - \mu_{r,j}|)$.

With this method, a user, could run SVTD without any previous knowledge on the feature to extract. This operation has an additional computational cost, as it requires to perform n times a $k \times k$ SVD; however we will see in the next remark that this cost has not a great impact on the total computational complexity, as it is dominated by the cost of other, more expensive, computations.

Remark 3.2 (Complexity analysis)**.** We start analyzing the time complexity. Using randomized SVD techniques (see Halko et al., 2011), step 1 can be carried out with a total of $O(n^2k)$ steps, the SVD on $E^\dagger M_{3,r}(E^\top)^\dagger$ requires $O(k^3)$ steps while step 6 requires $O(n^2k)$ steps for matrix multiplication for each one of the n iterations. So the overall complexity of the Algorithm 1 is thus

$$O(n^2k + k^3 + n^3k).$$

To this, we should add the additional computational cost of the feature-selection method outlined in Remark 3.1: that method requires to perform n times a $k \times k$ SVD, costing $O(nk^3)$; however, as $n \geq k$, this cost is dominated by the $O(n^3k)$ component. It is important to highlight that the implementation described in Algorithm 1 has mainly a descriptive purpose; for a specific LVM, optimized implementations may exists. For the single-topic model, for example, the method can be implemented without never calculating explicitly the tensor M_3, calculating in one step its whitened slices H_r, with a complexity of $O(Nnk)$ and then performing the subsequent diagonalizations in $O(n^2k^2)$ time. Additional tuning of the performances can be obtained exploiting the sparsity of the data, using for matrix operations sparse matrix technique. We also remark that the algorithm is trivially parallelizable: assuming we have m machines on which to parallelize steps 5, 6, 7 of the algorithm and the feature selection task, we can reduce the total running time to

$$O(n^2k + k^3 + \frac{n^3k}{m}).$$

Regarding memory, notice that we never use the full tensor M_3, but only its $r - th$ slice; the overall memory complexity of the algorithm is thus $O(n^2)$.

These complexity requirements are comparable to the ones of Anandkumar et al. (2012a,b); however, these methods are randomized, with nontrivial variance in their output, so they may require several runs of the full algorithm in order to provide accurate results. Tensor power method from Anandkumar et al. (2014) has in general a worst computational complexity: it is an iterative technique, with a number of iterations difficult to bound a priori; the authors suggest that accuracy ϵ can be reached with $O(k^{5+\delta}(\log(k) + \log\log(1/\epsilon)))$ operations, among iterations, random restarts and actual matrix operations, compared to our $O(k^3)$; to this time we need to add the time necessary to get the $k \times k \times k$ tensor from the sample, a computational time that is not trivial for many LVMs.

Remark 3.3 (On the generality of the algorithm)**.** We remark that during the construction of the algorithm we have not made any hypotheses on the probability distribution of the data; instead, we have required the matrix M to be full rank, with at least one feature r with different conditional expectations on the various topics. In the real world applications this is a realistic requirement: for example, in the topic modeling case, it means that we need at least one word whose probability of appearing is not exactly the same among the various topics. Also, we do not need to know in advance what this word is, as Remark 3.1 explains. This requirement is not present in the other matrix-based methods, as they rely on a randomized matrix to guarantee the uniqueness of the results (see Remark 3.4); however the learning performances of these methods are significantly poorer than those of SVTD, as Section 5 shows. It is an interesting open problem to find a deterministic

method joining the scalability properties of simultaneous diagonalization methods, without requiring this separation condition.

Remark 3.4 (Alternative algorithms)**.** As said in the introduction, other algorithms exist to retrieve the unknowns (M, ω) from M_1, M_2 and M_3. The most popular way is TPM, described in (Anandkumar et al., 2014). In that algorithm, the idea is to find a matrix W (it may be, for example, the pseudoinverse of the whitening matrix E), such that $W^\top M_2 W = I$, and use it to whiten the tensor M_3, getting a $k \times k \times k$ tensor T, from whose robust eigenvectors it is possible to retrieve the model parameters. To get the set of the robust eigenvectors, the authors use a three-dimensional extension of the well-known matrix power method. While very robust, the implementation of this method may result complex for who is not familiar with tensors; in addition, it is an iterative method, and so it requires a tuning of the hyperconvergence parameters, that might require many trial-and-error tests. This practical considerations, together with the high computational complexity, as outlined in Remark 3.2, are drawbacks that matrix methods do not have.

Matrix methods, or "simultaneous diagonalization" methods, as those outlined in (Anandkumar et al., 2012b) and (Anandkumar et al., 2012a), are technically more similar to the method presented here, and they are two variations of the same approach, one, that in (Anandkumar et al., 2012a), using eigenvectors, and the other in (Anandkumar et al., 2012b) using singular vectors. Those methods take a random vector $\eta \in \mathbb{R}^n$, and observe that the matrix $M_3(\eta)$, defined as

$$(M_3(\eta))_{i,j} := \sum_{l=1}^{n}((M_3)_{i,j,l}\eta_l)$$

can be decomposed as follows:

$$M_3(\eta) = M\Omega^{1/2} diag((\eta\mu_1^\top, ..., \eta\mu_k^\top))\Omega^{1/2}M^\top.$$

Then, they calculate the whitening matrix E and get the matrix

$$H_\eta = E^\dagger M_3(\eta)(E^\dagger)^\top$$

from whose left singular vectors O they retrieve M (up to rescaling and columns reordering) solving the following system of equations:

$$\begin{cases} M\Omega^{1/2} = EO \\ M\omega = M_1 \end{cases}$$

using essentially Equation (7). The introduction of the random vector η has the scope of guaranteeing that the elements of $(\eta\mu_1^\top, ..., \eta\mu_k^\top)$ are almost surely distinct, and so O is unique. So, we can see that there are essentially two main differences: the first is the fact that instead of using a randomized matrix, we fix a specific feature r, choosing the one with the maximum minimum variation between the feature components; in particular, this is the same of saying that we fix η to be the $r-th$ coordinate vector, providing a recipe for finding r in Remark 3.1. In this way we provide the choice that maximizes the stability. The second difference is the fact that we do not retrieve the matrix M from Equation (7), but observing that, if η_i is the $i-th$ coordinate vector, then

$$M_3(\eta_i) = M\Omega^{1/2} diag(m_i)\Omega^{1/2}M^\top.$$

and so, for each $i = 1, ..., n$, we can find the row m_i of M as the singular values of the various $H_i = E^\dagger M_3(\eta_i)(E^\dagger)^\top$. In this sense, our method relies on the singular *values* of a SVD decomposition

and not on the singular vectors. In our experiments, see Section 5, we found this approach much more stable if compared with other methods.

A different line of work, instead of using matrix decomposition techniques to perform simultaneous diagonalization, relies on matrix optimization methods (Kuleshov et al., 2015; Colombo and Vlassis, 2016). These methods recover the matrix M by finding the matrix that jointly diagonalizes/triangularize the slice of M_3, solving an optimization problem; however, guarantees to optimally solve this problem are provided only in limited settings; furthermore, these methods suffer of poor computational performances if compared with the methods listed above (decomposing a $100 \times 100 \times 100$ rank-8 tensor with the method from Kuleshov et al. (2015), using the authors' matlab implementation, took around 15 minutes, while it took less than 1 second with SVTD).

4 Perturbation Analysis

In the previous section we have outlined an algorithm that accurately learns the model parameters of a LVM, given the exact values of M_1, M_2 and M_3. However, when applying our algorithm to a real world problem, we never have these exact variables, but only a set of estimators $\tilde{M}_1$, $\tilde{M}_2$ and $\tilde{M}_3$ that are expected to become arbitrarily accurate as sample size increases. This fact has some immediate consequences: first we need to adapt SVTD to deal with perturbed matrices (as $\tilde{M}_1$, $\tilde{M}_2$ and $\tilde{M}_3$ are) that do not necessarily have rank k or are not assured to be positive definite. Second, we need to study how the perturbations on those estimates propagate up to the final results. The adaptation of SVTD is outlined in Algorithm 2.

Algorithm 2 SVTD when $\tilde{M}_1$, $\tilde{M}_2$ and $\tilde{M}_3$ are available, instead of M_1, M_2 and M_3.

Require: $\tilde{M}_1, \tilde{M}_2, \tilde{M}_3$, and the number of latent states k

1: Decompose $\tilde{M}_2$ as $\tilde{M}_2 \approx \tilde{U}_k \tilde{S}_k \tilde{V}_k$ with a SVD truncated at the k-th singular vector.
2: Define the whitening matrix $\tilde{E} = \tilde{U}_k {\tilde{S}_k}^{1/2}$ and calculate its pseudoinverse $\tilde{E}^\dagger = (\tilde{S}_k)^{-1/2} {\tilde{U}_k}^\top$.
3: Select a feature r and compute $\tilde{M}_{3,r}$
4: Compute $\tilde{O}$ as the left singular vectors of $\tilde{H}_r := \tilde{E}^\dagger \tilde{M}_{3,r} (\tilde{E}^\dagger)^\top$
and $\tilde{m}_r$ as the singular values.
5: **for** $i = 1 \to n$ **do**
6: Compute $\tilde{H}_i := \tilde{E}^\dagger \tilde{M}_{3,i} (\tilde{E}^\dagger)^\top$
7: Obtain the $i-$th row of $\tilde{M}$ as the diagonal entries of $\tilde{O}^\top \tilde{H}_i \tilde{O}$
8: **end for**
9: Obtain $\tilde{\omega}$ solving $\tilde{M}_1 = \tilde{M}\tilde{\omega}$
10: Return $(\tilde{M}, \tilde{\omega})$

The modifications with respect to Algorithm 1 need to guarantee that we deal with positive definite matrices with rank k. We do this by performing the whitening in steps 4 and 6, using the product of the left singular vectors of $\tilde{M}_2$ and the first k singular values and, in step 4, taking $\tilde{O}$ to be the left singular vectors of $\tilde{H}_r$; the rest of the algorithm is identical.

Remark 4.1. In step 1 of Algorithm 2 we obtain a rank k approximation of $\tilde{M}_2$ as $\tilde{M}_2 \approx \tilde{U}_k \tilde{S}_k \tilde{V}_k$. In general, as $\tilde{M}_2$ converges to M_2 which is positive semidefinite, we expect that for suitably big

A python implementation of this algorithm can be found in `https://github.com/mruffini/SpectralMethod.git`.

samples $\tilde{V}_k = \tilde{U}_k^\top$, i.e. that $\tilde{M}_2$ is also positive semidefinite. The size of the sample required to have $\tilde{M}_2$ positive semidefinite depends on the concentration properties of the sample. However, Algorithm 2 produces accurate results even when positive definiteness of $\tilde{M}_2$ is not guaranteed, as explained next in Theorem 4.1.

We now study the accuracy of the algorithm. Intuitively, the more similar the perturbed $\tilde{M}_1$, $\tilde{M}_2$ and $\tilde{M}_3$ are to the exact M_1, M_2 and M_3, the better the outcomes of the algorithm should be. This intuition is confirmed by the following theorem.

Theorem 4.1. Given the unperturbed versions of of M_1, M_2 and M_3 and the feature we want to isolate, r, let α_r and α_{M_2} be

$$\alpha_r = \min_{i \neq j}(|\mu_{r,i} - \mu_{r,j}|) > 0, \quad \alpha_{M_2} = \min_{i \leq k}(\sigma_i(M_2)^2 - \sigma_{i+1}(M_2)^2) > 0$$

where $\sigma_i(M_2)$ are the singular values of M_2. Assume the empirical estimates $\tilde{M}_2$ and $\tilde{M}_3$ satisfy

$$||\tilde{M}_2 - M_2||_F < \epsilon, \quad ||\tilde{M}_3 - M_3||_F < \epsilon.$$

Then, there exists a function $\gamma(M, \omega)$, of the model parameters, such that, if $\epsilon < \gamma(M, \omega)$, Algorithm (2), fed with $\tilde{M}_1$, $\tilde{M}_2$ and $\tilde{M}_3$, provides an estimated matrix $\tilde{M}$ whose rows $\tilde{m}_i$, for all $i \in \{1, ..., n\}$, satisfy

$$||m_i - \tilde{m}_i||_2 \leq C_1\epsilon + \sqrt{k}\frac{\epsilon}{\alpha_r}(C_2 + \frac{C_3}{\alpha_{M_2}}) + O(\sqrt{k}C_4\epsilon^2)$$

where m_i are the rows of M, C_1, C_2 and C_3 are polynomial functions of $||E||_F, ||O||_F$ and $||M_{3,i}||_F$, and C_4 is a polynomial function of $||E||_F, ||O||_F, ||M_{3,i}||_F, \frac{1}{\alpha_r}$ and $\frac{1}{\alpha_{M_2}}$.

Note the key role of α_r, the minimum difference between the elements of $(\mu_{r,1}, ..., \mu_{r,k})$; when samples are large enough, the theorem guarantees that the algorithm works correctly, although "large enough" depends on α_r. When this condition is not satisfied, the learning algorithm still works, but might provide output results that are different from the theoretical generative model. We recall here Remark 3.1, where we wondered how to select the proper feature r; ideally, the one that would guarantee the highest accuracy would be the one with the highest possible α_r.

Theorem 4.1, together with Theorem 2.2, (resp. Theorem 2.3), provides a sample complexity bound for the Single Topic Model (resp. for LDA). For any given accuracy that ones wants to obtain in the estimates of the parameters (M, ω), using Theorems 2.2 and 4.1 one can understand the sample size needed to reach that accuracy with high probability.

5 Experiments

In this section we will provide some experiments, to test both on synthetic and real data, the results we presented in this paper.

5.1 Recovering M_2 and M_3

In Section 2.1 we described a new estimator to recover the matrix M_2 and tensor M_3 from a sample, comparing it with the methods presented in the state of the art literature from Zou et al.

For an explicit formulation of the value of $\gamma(M, \omega)$, we refer the reader to the proof of the theorem and to Remark B.1.

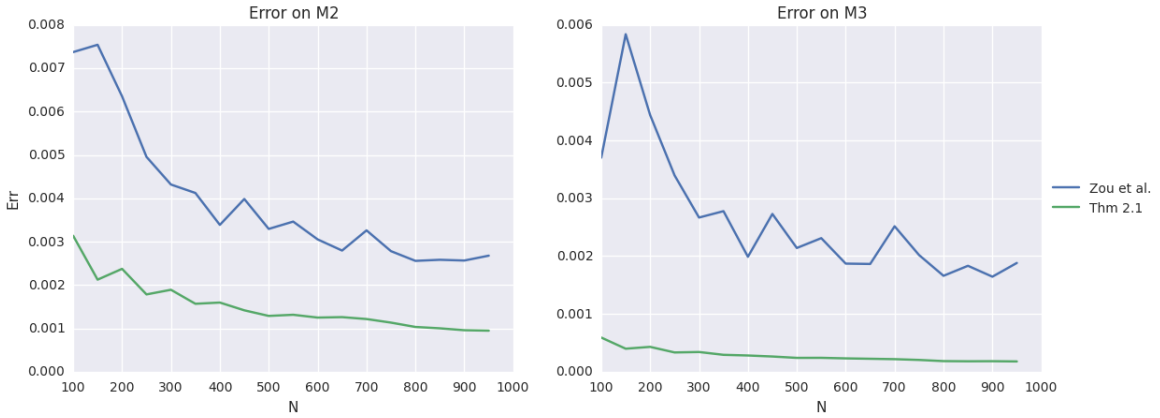


Figure 1: The $x-$axis of the figures represents the sizes of the synthetic text corpora, wile the $y-$axis is Err_2 for the left chart and Err_3 for the right chart. Green lines represent the errors obtained with the method presented in Theorem 2.1, while blue lines represent the errors obtained with the method by Zou et al. (2013) on the same corpora.

.

(2013), outlined in Remark 2.3. In this section we compare, using synthetically generated data, how well the two different methods recover M_2 and M_3 as a function of the sample size. To perform this experiment, we generated a set of 1000 synthetic corpora according to the single topic model described in Section 2.1, with different sizes (the number of texts for each corpus); the smallest corpus contained 100 texts, the largest 10000; each text contained a random number of words, from a minimum of 3 to a maximum of 100. For each corpus, the values of the unknowns (M, ω) have been randomly generated, and from them we have been able to obtain the theoretical values of M_2 and M_3 using equations (3) and (4) and to compare those values with the one empirically estimated from data using the equations in Theorem 2.1 for the presented method and the method from Zou et al. (2013) for the competing one. Results appear in Figure 1, where we show how the estimated M_2 and M_3, say $\tilde{M}_2$ and $\tilde{M}_3$, approach the theoretical values; in particular, in the chart are represented the errors

$$Err_2 = ||\tilde{M}_2 - M_2||_F, \quad Err_3 = ||\tilde{M}_3 - M_3||_F$$

as a function of the sample size N used to find $\tilde{M}_2$ and $\tilde{M}_2$. We can see that the method of Theorem 2.1 outperforms the state of the art technique; this is due to the fact that it gives more weight to the longer documents, where the signal is more clear, and less to the shorter, where the signal is noisier.

5.2 Recovering (M, ω) from a random sample

We now test the ability of SVTD to recover the unknown parameters from random set of data, comparing it with state of the art methods. We fix a dictionary of $n = 100$ words with $k = 5$ topics and we generate a sample distributed as a Single Topic Model: for various sizes comprised between $N = 50$ and $N = 1000$ we generate synthetic corpora and we use them to learn the model parameters. For each sample corpus we proceed as follows:

- We estimate the values of $\tilde{M}_1, \tilde{M}_2$ and $\tilde{M}_3$ using Theorem 2.1.

- We retrieve from the estimated $\tilde{M}_1, \tilde{M}_2$ and $\tilde{M}_3$ the pair of unknowns $(\tilde{M}, \tilde{\omega})$ using SVTD as in Algorithm 2. Also we generate alternative solutions using the decomposition algorithms from Anandkumar et al. (2014) ("Tensor power method"), from Anandkumar et al. (2012a) ("Eigendecomposition method") and from Anandkumar et al. (2012b) ("SVD method"), that are the current reference methods.

- Each time we generate a solution, we register the time in seconds employed by the various algorithms. For each method, we represent the average time employed to perform the parameters recovery along the various runs in Figure 2a.

- For each set of retrieved parameters $(\tilde{M}, \tilde{\omega})$ we calculate the learning error as follows:

$$Err = || \sum_{i=1}^{k} \tilde{\omega}_i \tilde{\mu}_i \otimes \tilde{\mu}_i \otimes \tilde{\mu}_i - M_3||_F$$

  where $\tilde{\mu}_i$ are the columns of $\tilde{M}$, and where M_3 is obtained from the true parameters of the corpus (M, ω) as in Equation (4). For each sample size, we repeat the experiment 10 times, and we plot in Figure 2b the results of the analysis as a function of N.

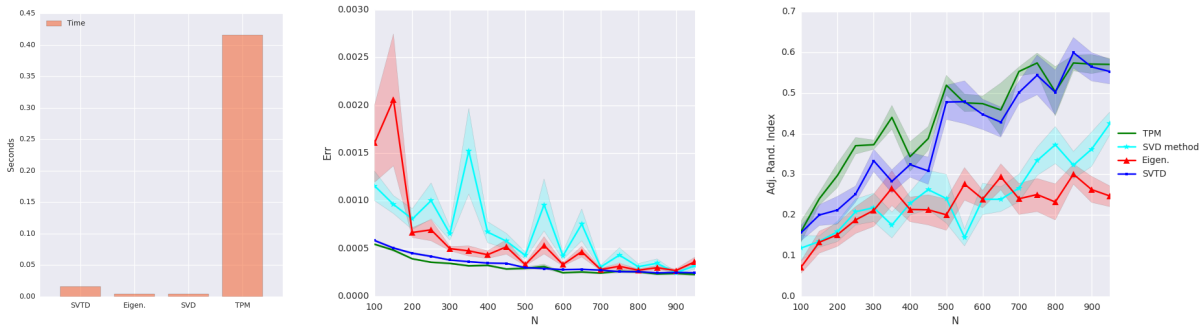
- For each run of the experiment and for each of the methods, we use the retrieved parameters to cluster the various documents of the considered corpus, using the MAP assignment described in Remark 2.1. We compare the clustering accuracy of the various methods using the Adjusted Rand Index (Hubert and Arabie, 1985) of the partition of the data obtained with MAP assignment with the one obtained using the true topics. For each sample size, we repeat the experiment 10 times, and we plot in Figure 2c the results of the analysis as a function of N.

In Figure 2a, the average running times of the various methods are presented. As expected, matrix-based methods preform similarly, and work much faster than TPM, as a consequence of the better dependence on the number of latent states. SVTD has a slightly larger running time, due to the feature selection process outlined in Remark 3.1. Looking at Figures 2b and 2c, we can see that the learning accuracy of SVTD is comparable to that of TPM, in terms of both reconstruction error and clustering accuracy; SVTD and TPM outperforms the other matrix based method, which provide very poor performances of the two tasks. Peaks in the error are present when a method fails to provide reliable results (for example, a matrix $\tilde{M}$ very far from the simplex).

We are thus able to conclude that SVTD provides a learning accuracy similar to TPM. Both the methods outperform in term of accuracy existing matrix methods, but while TPM has significantly larger running times, SVTD shares with the other matrix-based methods competitive computational performances.

Practical considerations.

In linear algebra operations, the running time is highly influenced by many implementation details, such as the usage of vectorized operations. All the algorithms have been implemented in Python 2.7, using *numpy* (Van Der Walt et al., 2011) library for linear algebra operations. All the experiments have run on a MacBook Pro, with an Intel Core i5 processor.



(a) (b) (c)

Figure 2: In Figure 2a the average running times are represented; TPM has significantly longer running time, due to the worst dependence on the number of latent states, while matrix methods behave similarly. Figures 2b and 2c contain the analysis of the learning accuracy; in both cases the $x-$axis represents the size of the synthetic text corpora. In 2b the $y-$axis is Err, the reconstruction error for the various tested methods, while in Figure 2c the $y-$axis is the clustering accuracy. Shaded area represent the variance of the output of the experiments over 10 runs with same sample size: SVTD (blue line) behaves in a very similar way to tensor power method, outperforming SVD and Eigendecomposition method.

5.3 Real data

To perform experiments with real data, we analyzed two different corpora: the list of State of the Union addresses since 1945 to 2005 and Dante's "Divina Commedia". In both cases, we used a dictionary of $n = 3000$ words, so the data matrix X was a $N \times 3000$, where N was the size of the corpus (we had $N = 65$ on the first example and $N = 100$ in the second), while tensor M_3 belonged to $\mathbb{R}^{3000\times3000\times3000}$. We deliberately used corpora with $N << n$, in order to test the ability of our algorithm to work with small sets of documents.

5.3.1 Dante's Divina Commedia

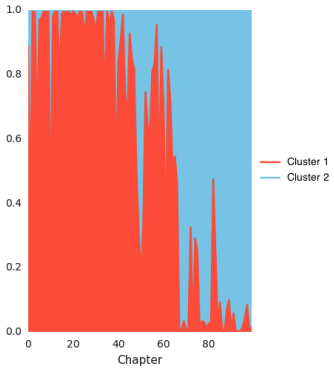
Dante's "Divina Commedia" (Alighieri, 1979) is an Italian epic poem written in the first half of 14th century; it deals with the imaginary trip of the main character, Dante, in the afterlife, guided by Virgilio, the famous Latin poet, and Beatrice, a Florentine woman that inspired most of Dante's works. The story line represents an allegorical description of death soul's journey towards God according to medieval world view. It begins with Dante's travel trough the "Inferno" (Hell), where damned souls are deemed to eternal punishment according to their sins; the journey then moves to "Purgatorio", a seven level mountain, where, at each level, a capital sin (sins less serious than those punished in Hell) is allegorically described; here souls are discounting their punishment, before finally move to "Paradiso", Heaven, that is visited by Dante in the last third of the book. The book is made of 100 different chapters: 34 for Hell, 33 for Purgatory and 33 for Heaven.

Single Topic Model

The full text can be found here: `http://www.gutenberg.org/files/1012/1012-0.txt`

Cluster	Most representative words	Assigned chapters
1	fosso, bolgia, scoglio, avante, coda, grido, rotta, ponte, stanchi, maestro	Inferno (1,3-10, 12-34), Purgatorio (1-8, 10-12,20-24,26,27,29)
2	milizia, segue, intende, letizia, conosce, lumi, ama, Beatrice, piacer, Cristo	Inferno (2,11) Paradiso (1-33), Purgatorio (6, 13-19, 25, 28, 30-33)

(a)



(b)

Figure 3: Fig. 3a: the cluster assignment for "Divina Commedia" chapters. In Figure 3b, the $x-$axis represents the chapter of the book: the first 34 are Hell, from 35 to 67 we have Purgatory and the last 33 are Heaven. The red area represents how much each chapter belongs to the first cluster, while the blue area how much it belongs to second cluster.

We run SVTD for the Single Topic Model on the "Divina Commedia" corpus, on the $N = 100$ texts, represented by the 100 chapters of the book. We tested various possible number of topics, but surprisingly almost always the algorithm produced two significant topics. In the table 3a we can see the results of the algorithm run with $k = 2$: for each of the two topics a cluster has been defined; the most representative words are represented together with the chapters assigned to each cluster. We can see that most of the chapters of Hell are assigned to the same cluster; also the chapters of Heaven are all assigned to the same cluster while Purgatory is assigned in part to cluster 1 and in part to cluster 2.

Latent Dirichlet Allocation

As a second step, we used the same corpus, with the same vocabulary, to infer for each chapter how much it deals with topic 1 and how much it deals with topic 2, using Latent Dirichlet Allocation. In particular, we retrieved from the data the tensors $\tilde{M}_2^\alpha$ and $\tilde{M}_3^\alpha$ from Theorem 2.3 and we used them to feed Algorithm 2. We had to specify a value for α_0, that was set to 2. With Algorithm 2 we retrieved a pair (M, α). We then used this pair and partially collapsed Gibbs Sampling to infer the topic mixture for each chapter. As we just have two topics, it is easy to plot the results of that inference, see Figure 3b. In this chart, the $x-$axis represents the progression of the chapters along the book, while red and blue stacked areas represent the value of the topic proportions for each chapter. We can see that most of the first 34 chapters have a strong predominance of the first topic, marked with the red area, that consequently can be identified with the Hell topic. In the same way, second topic, or Heaven, is very strong in the last 33 chapters. Purgatory, in the middle part of the plot, has a mixed belonging. The interesting fact is that the proportion of the Heaven topic seems to increase as the chapters approach the Heaven section, corresponding to Dante's ascent of the Purgatory mountain.

5.3.2 State of the Union addresses

Each year, the president of United States of America presents a speech to a joint session of the United States Congress, where he outlines his governative agenda, the national priorities and legislative projects. We considered the set of $N = 65$ state of the union addresses presented between 1945 and 2005, and we applied to this corpus the algorithm for the Single Topic Model, with the purpose of finding the most representative topics of the corpus and clustering the various speeches according to the learned topics. We run SVTD assuming to have $k = 5$ different topics, although with different values of k results were similar; we then grouped the speeches assigning them to the topic with the highest likelihood, using the Bayesian posterior assignment of Remark 2.1. In Table 1 we can see the results of this operation. For each topic a cluster has been defined; the most representative words are represented together with the speeches assigned to each cluster; for each president, the brackets indicate the year of the speech.

Cluster	Most representative words	Assigned speeches
1	construction, fiscal, legislative, peacetime, facilities, recommendations, projects, existing, transportation, veterans	Truman (1946 to 1950), Eisenhower (1953 to 1957, 1959), Kennedy (1961 to 1963), Johnson (1966,1967), Nixon (1973), Ford (1975,1977)
2	reducing, regulations, recovery, taxpayers, market, bills, weeks, nothing, gone, productivity	Johnson (1963 to 1965, 1968, 1969), Nixon (1970 to 1972, 1974), Ford (1976), Carter (1978), Reagan (1981 to 1988), Bush (1990, 1992)
3	ideals, soviet, missile, potential, missiles, world, conflict, struggle, countries, threat	Truman (1945, 1951), Eisenhower (1958, 1960), Carter (1979, 1980), Bush (1991a,1991b), G.W. Bush (2003)
4	companies, invest, 21st, teachers, parents, revolution, lowest, challenge, credit, bipartisan	Bush (1989), Clinton (1993 to 2000)
5	September, enemies, terror, compassion, terrorists, Afghanistan, relief, retirement, Iraq, dangerous	G.W. Bush (2001a, 2001b, 2002, 2004, 2005)

Table 1: The cluster assignment for the State of the Union addresses.

Cluster 1 contains only speeches of the Cold War period. Words like 'peacetime', 'construction' and 'projects' characterize, for instance, the period after the second World War, with the Marshall plan and Truman doctrine. Cluster 2 has speeches that belong to a wider set of dates, but all seem to be about internal politics and economics; among the most representative words we can find 'taxpayers', 'regulations' and 'productivity', key themes of the Reagan administration. Cluster 3 is clearly related to war with words as 'missile', 'conflict' and 'struggle'; the speeches come from presidents involved in important wars (WWII and Gulf wars). Cluster 4 mainly contains Bill Clinton

The full corpus can be found in link `http://www.nltk.org/nltk_data`.

speeches and has words that characterize the economic expansion of the 90ies ('companies', 'invest', 'credit'). Cluster 5 has among the top words 'terror', 'Afghanistan' and 'Iraq'; they all reveal post 9/11 politics carried on by G.W. Bush.

6 Conclusion and Future Work

We described a simple algorithm to learn latent variable models in polynomial time which shares many good characteristics of previous spectral methods, having at least one advantage over each of them (be in efficiency, or being in learning accuracy); together with this, we have introduced an efficient method for estimating the symmetric tensors of the moments for the Single Topic Model and for LDA.

A natural future work is to adapt this algorithm to an on-line, streaming environment (Liberty, 2013; Jain et al., 2016). In the theoretical front, we want to improve the perturbation Theorem 4.1, removing the dependence from α and α_M. On the applications side, we are interested in applying this algorithm to learn LVM in the healthcare analytics field, for instance to construct disease progression models and patient clusterings. Genetic data, where e.g. one typically has many more genes or SNPs than sequenced individuals, would also be of interest.

Acknowledgements

M. Casanellas is is partially funded by AGAUR project 2014 SGR-634 and MINECO/FEDER project MTM2015-69135-P. R. Gavaldà is partially funded by AGAUR project 2014 SGR-890 (MACDA) and by MINECO project TIN2014-57226-P (APCOM).

Appendix

A Proofs for Section 2

A.1 Proof of Theorem 2.1

Proof. We will prove the statements only for

$$\mathbb{E}(\frac{\sum_{i=1}^{N}(X^{(i)})_h(X^{(i)})_l}{\sum_{i=1}^{N}(c_i-1)c_i}) = \sum_{j=1}^{k}\omega_j\mu_{h,j}\mu_{l,j}.$$

Similar arguments hold for the other equations. It is easy to see, by conditional independence, that

$$E((X^{(i)})_h(X^{(i)})_l) = \sum_{j=1}^{k}\omega_j E((X^{(i)})_h(X^{(i)})_l|Y=j)$$

but the conditioned $(X^{(i)})_h$ and $(X^{(i)})_l$ are components of a multinomial distribution and so

$$\sum_{j=1}^{k}\omega_j E((X^{(i)})_h(X^{(i)})_l|Y=j) = \sum_{j=1}^{k}\omega_j(c_i^2-c_i)\mu_{h,j}\mu_{l,j}$$

which implies the thesis. □

A.2 Proof of Theorem 2.2

Proof. We want to express in a suitable way the elements of the matrix

$$\tilde{M}_2 - M_2 \tag{8}$$

and then express a bound using McDiarmid's inequality (McDiarmid, 1989). We know by construction that, for any $i \in \{1, ..., N\}$ it holds that

$$X^{(i)} = \sum_{j=1}^{c_i} x_j^{(i)}$$

where each $x_j^{(i)}$ is the $j-$th word of the document. We thus consider the set of all the words from all the documents:

$$\mathcal{X} = (x_1^{(1)}, ..., x_{c_1}^{(1)}, ..., x_1^{(N)}, ..., x_{c_N}^{(N)}).$$

It is easy to see that $\tilde{M}_2$ can be expressed as a function of $\mathcal{X}$, for all pairs $u, v \in \{1, ..., n\}$ we have

$$(\tilde{M}_2)_{u,v}(\mathcal{X}) = \frac{\sum_{i \neq j}(x_i^{(1)})_u (x_j^{(1)})_v + ... + \sum_{i \neq j}(x_i^{(N)})_u (x_j^{(N)})_v}{C_2}$$

where

$$C_2 = \sum_{i=1}^{N} c_i(c_i - 1).$$

We now define the following function:

$$\Phi(\mathcal{X}) = ||\tilde{M}_2(\mathcal{X}) - M_2||_F$$

and observe that, given

$$\mathcal{X} = (x_1^{(1)}, ..., x_{c_1}^{(1)}, ..., x_i^{(l)}, ..., x_1^{(N)}, ..., x_{c_N}^{(M)})$$

and

$$\mathcal{X}^\top = (x_1^{(1)}, ..., x_{c_1}^{(1)}, ..., {x_i^{(l)}}^\top, ..., x_1^{(N)}, ..., x_{c_N}^{(M)})$$

we have

$$|\Phi(\mathcal{X}) - \Phi(\mathcal{X}^\top)| \leq ||\tilde{M}_2(\mathcal{X}) - \tilde{M}_2(\mathcal{X}^\top)||_F =$$

$$= \sqrt{\sum_{u,v=1}^{n} \left(\frac{\sum_{i \neq j}(x_j^{(l)})_u ((x_i^{(l)})_v - ({x_i^{(l)}}^\top)_v)}{C_2} \right)^2} \leq \frac{\sqrt{2}\max_j(c_j)}{C_2}.$$

We are now able to apply McDiarmid's inequality stating that

$$\mathbb{P}(||\tilde{M}_2 - M_2||_F > \mathbb{E}(||\tilde{M}_2 - M_2||_F) + \epsilon) \leq e^{-\frac{\epsilon^2 C_2^2}{(\max_j(c_j))^2 C_1}}.$$

So, by setting

$$t = \frac{\epsilon C_2}{\max_j(c_j)\sqrt{C_1}}$$

we get

$$\mathbb{P}(||\tilde{M}_2 - M_2||_F > \mathbb{E}(||\tilde{M}_2 - M_2||_F) + t\frac{\max_j(c_j)\sqrt{C_1}}{C_2}) \leq e^{-t^2}.$$

We now provide a bound for $\mathbb{E}(||\tilde{M}_2 - M_2||_F)$. We begin observing that

$$\tilde{M}_2 = \sum_{i=1}^{N} w_i \tilde{M_2}^{(i)}$$

where $w_i = \frac{c_i(c_i-1)}{C_2}$ and $\tilde{M_2}^{(i)}$ are independent matrices defined as follows:

$$(\tilde{M_2}^{(i)})_{(u,v)} = \frac{\sum_{l\neq j}(x_l^{(i)})_u(x_j^{(i)})_v}{c_i(c_i-1)}$$

Notice that, for any i, $\mathbb{E}(\tilde{M_2}^{(i)}) = M_2$. Using Jensen's inequality we have

$$\mathbb{E}(||\tilde{M}_2 - M_2||_F) \leq \sqrt{\mathbb{E}(||\tilde{M}_2 - M_2||_F^2)}.$$

This last term is equal to

$$\sqrt{\mathbb{E}(||\tilde{M}_2||_F^2) - ||M_2||_F^2} = \sqrt{\sum_{u,v}\mathbb{E}((\sum_{i=1}^{N} w_i(\tilde{M_2}^{(i)})_{(u,v)})^2) - ||M_2||_F^2} =$$

$$= \sqrt{\sum_{u,v}\mathbb{E}(\sum_{i=1}^{N} w_i^2(\tilde{M_2}^{(i)})_{(u,v)}^2) + \sum_{u,v}\mathbb{E}(\sum_{i\neq j} w_j w_i(\tilde{M_2}^{(i)})_{(u,v)}(\tilde{M_2}^{(j)})_{(u,v)}) - ||M_2||_F^2}$$

and using the fact that $\mathbb{E}(\tilde{M_2}^{(i)}_{(u,v)}\tilde{M_2}^{(j)}_{(u,v)}) = (M_2)^2_{(u,v)}$, this equals

$$\sqrt{\sum_{u,v}\sum_{i=1}^{N} w_i^2\mathbb{E}(||\tilde{M_2}^{(i)}||_F^2) + \sum_{i\neq j} w_j w_i||M_2||_F^2 - ||M_2||_F^2}\,.$$

Now using that $||\tilde{M_2}^{(i)}||_F \leq 1$, we can bound this from above by

$$\sqrt{\sum_{i=1}^{N} w_i^2 + ||M_2||_F^2(\sum_{i\neq j} w_j w_i - 1)} = \sqrt{\sum_{i=1}^{N} w_i^2(1 - ||M_2||_F^2)}.$$

where in the last equality we used the fact that $\sum_{i\neq j} w_j w_i = 1 - \sum_{i=1}^{N} w_i^2$. So, if we call $W_2^{(N)} = \sum_{i=1}^{N} w_i^2$, we have $\mathbb{E}(||\tilde{M}_2 - M_2||_F) \leq \sqrt{W_2^{(N)}(1 - ||M_2||_F^2)}$, from which we obtain

$$\mathbb{P}(||\tilde{M}_2 - M_2||_F > \sqrt{W_2^{(N)}(1 - ||M_2||_F^2)} + t\frac{\max_j(c_j)\sqrt{C_1}}{C_2}) \leq e^{-t^2}.$$

In conclusion, we can state that if $e^{-t^2} = \delta$ we get, for any $\delta \in (0, 1]$

$$\mathbb{P}(||\tilde{M}_2 - M_2||_F > \epsilon) \leq \delta$$

where

$$\epsilon = \sqrt{W_2^{(N)}(1 - ||M_2||_F^2)} + \sqrt{\log(\frac{1}{\delta})}\frac{\max_j(c_j)\sqrt{C_1}}{C_2}.$$

A similar argument works for M_3. □

B Proofs for Section 4

B.1 Proof of Theorem 4.1

Proof. The goal of the proof is to develop a perturbation bound for each row m_i of the unknown matrix M such that, $||\tilde{m}_i - m_i||_2 \leq Bound(\epsilon)$, for a certain function $Bound(\epsilon)$. We notice, from Algorithm 2, that each $\tilde{m}_i$ is obtained as the diagonal entries of the following matrix:

$$\tilde{O}^\top \tilde{E}^\dagger \tilde{M}_{3,i} (\tilde{E}^\top)^\dagger \tilde{O}$$

and so, we will need to find the perturbations of the matrices composing this equation, as the following relation holds:

$$||\tilde{m}_i - m_i||_2 \leq ||\tilde{O}^\top \tilde{E}^\dagger \tilde{M}_{3,i} (\tilde{E}^\top)^\dagger \tilde{O} - O^\top E^\dagger M_{3,i} (E^\top)^\dagger O||_F.$$

In short, having perturbation bounds on $\tilde{M}_{3,i}$,$\tilde{E}^\dagger$ and $\tilde{O}$ will be sufficient to reach our goal.

Perturbations on $\tilde{M}_{3,i}$

We know, by hypothesis of the theorem, that $||\tilde{M}_{3,i} - M_{3,i}||_F = ||\Delta_{M_{3,i}}||_F < \epsilon$.

Perturbations on $\tilde{E}^\dagger$

It is a known fact (see Stewart, 1990) that, given the SVD $\tilde{M}_2 = \tilde{U}\tilde{S}\tilde{V}$, and $M_2 = USU^\top$, if $||\tilde{M}_2 - M_2||_F < \epsilon$, we have that

$$||\tilde{S} - S||_F \leq \epsilon. \tag{9}$$

Algorithm 2 considers at step 2 the following approximation of E: $\tilde{E} = \tilde{U}_k(\tilde{S}_k)^{1/2}$, while the unperturbed value of E can be found as $E = U_k(S_k)^{1/2}$, where the subscript k indicates the truncation at the $k - th$ singular value. So, to reach a perturbation bound on $\tilde{E}^\dagger$, we first need to look for a perturbation bound on $\tilde{E}$, that will be obtained bounding the error of $(\tilde{S}_k)^{1/2}$ and $\tilde{U}_k$. The first one is a consequence of equation (9): if $\Delta_S = (\tilde{S}_k)^{1/2} - S_k^{1/2}$, we have

$$||\Delta_S||_F < \frac{\epsilon}{2\sqrt{\sigma_k(M_2)}},$$

where $\sigma_k(M_2)$ is the $k-$th the singular value of M_2. To find a bound on $\tilde{U}_k$ we will use Lemma B.1 to get

$$||\tilde{U}_k - U_k||_F < \sqrt{8k}\frac{2\epsilon||M_2||_F^2 + \epsilon^2}{\alpha_{M_2}}.$$

where $\alpha_{M_2} = \min_{i \leq k} (\sigma_i(M_2)^2 - \sigma_{i+1}(M_2)^2)$ and $\sigma_i(M_2)$ are the singular values of M_2. We thus conclude that, if $\Delta_U = \tilde{U}_k - U_k$, $\tilde{E} = E + \Delta_U S^{1/2} + U\Delta_S + \Delta_U \Delta_S$ and hence

$$||\tilde{E} - E||_F < f(\epsilon) = \epsilon \left(||S^{1/2}||_F \sqrt{8k} \frac{2||M_2||_F^2 + \epsilon}{\alpha_{M_2}} + \frac{||U||_F}{2\sqrt{\sigma_k(M_2)}} + \sqrt{8k} \frac{2\epsilon||M_2||_F^2 + \epsilon^2}{\alpha_{M_2}(2\sqrt{\sigma_k(M_2)})} \right).$$

We are now ready to find a perturbation bound on the pseudoinverse of $\tilde{E}$, using a known bound from (Stewart and guang Sun, 1990): if $||\tilde{E} - E||_F < f(\epsilon)$ then

$$||\tilde{E}^\dagger - E^\dagger||_F \leq f(\epsilon)\tau(E)$$

where $\tau(E) = ||E^\dagger||_F^2 + ||(E^\top E)^{-1}||_F ||\mathbb{I} - EE^\dagger||_F$.

Perturbations on $\tilde{O}$

We now look for the value of $g(\epsilon) = ||O - \tilde{O}||_F$. In particular, O comes from the decomposition of $E^\dagger M_{3,r}(E^\top)^\dagger$:

$$E^\dagger M_{3,r}(E^\top)^\dagger = O diag(m_r) O^\top. \tag{10}$$

while $\tilde{O}$ is the set of the left singular vectors of $\tilde{E}^\dagger \tilde{M_{3,r}} (\tilde{E}^\top)^\dagger$. First, we observe that

$$\begin{aligned} ||E^\dagger M_{3,r}(E^\top)^\dagger - \tilde{E}^\dagger \tilde{M_{3,r}} (\tilde{E}^\top)^\dagger||_F \leq \\ \leq 2f(\epsilon)\tau(E)||E^\dagger||_F ||M_{3,r}||_F + \epsilon||E^\dagger||_F^2 + \\ + 2\epsilon f(\epsilon)\tau(E)||E^\dagger||_F + f(\epsilon)^2\tau(E)^2(||M_{3,r}||_F + \epsilon) = h(\epsilon). \end{aligned} \tag{11}$$

Using Corollary B.1, we assume the hypothesis that

$$h(\epsilon) \leq \frac{\alpha_r^2}{\sqrt{2}\left(2||H_r||_F(1 + \sqrt{1 - \frac{1}{k}}) + \sqrt{\alpha_r^2 + 4||H_r||_F^2(1 + \sqrt{1 - \frac{1}{k}})^2}\right)} \tag{12}$$

where $H_r = E^\dagger M_{3,r}(E^\top)^\dagger$ and $\alpha_r = \min_{i \neq j}(|\mu_{r,i} - \mu_{r,j}|)$, to get that $||\tilde{O} - O||_F < 2\sqrt{2}\frac{h(\epsilon)}{\alpha_r} = g(\epsilon)$, We are now able to conclude our proof by analyzing

$$\begin{aligned} ||m_i - \tilde{m_i}||_2 \leq ||\tilde{O^\top} \tilde{E}^\dagger \tilde{M_{3,i}} (\tilde{E^\top})^\dagger \tilde{O} - O^\top E^\dagger M_{3,i}(E^\top)^\dagger O||_F \leq \\ \leq P_1 g(\epsilon) + P_2 \epsilon + P_3 f(\epsilon) + O(\epsilon^2 P_4), \end{aligned}$$

where P_1, P_2 and P_3 are polynomials in $||E||_F, ||O||_F$ and $||M_{3,i}||_F$, and P_4 is a polynomial in $||E||_F, ||O||_F, ||M_{3,i}||_F, g(\epsilon), \epsilon$ and $f(\epsilon)$. The thesis follows making explicit these polynomials.

□

Remark B.1. In the statement of Theorem 4.1, we said that there exists a number $\gamma(M, \omega)$, such that, if $\epsilon < \gamma(M, \omega)$, the perturbation bound of the thesis works. Looking at the proof of the theorem, we are able to explicitly calculate this number, just by solving the inequality (12). We can practically think at $\gamma(M, \omega)$ as the largest value of ϵ that satisfies this inequality.

Lemma B.1. *Consider M_2, the perturbed $\tilde{M}_2$, and their SVD, $\tilde{M}_2 = \tilde{U}\tilde{S}\tilde{V}^\top$, $M_2 = USU^\top$. Let $\tilde{U}_k$ and U_k be matrices of the first k left singular vectors of $\tilde{M}_2$ and M_2. Define $\alpha_{M_2} = \min_{i\leq k}(\sigma_i(M_2)^2 - \sigma_{i+1}(M_2)^2)$. If $||\tilde{M}_2 - M_2|| < \epsilon$, then $||\tilde{U}_k - U_k||_F < \sqrt{8k}\frac{2\epsilon||M_2||_F^2+\epsilon^2}{\alpha_{M_2}}$.*

Proof. Consider $\tilde{M}_2\tilde{M}_2{}^\top = \tilde{U}\tilde{S}^2\tilde{U^\top}$ and $M_2M_2^\top = US^2U^\top$. Then $||\tilde{M}_2\tilde{M}_2{}^\top - M_2M_2{}^\top||_F \leq 2\epsilon||M_2||_F^2 + \epsilon^2$. Take now the matrix of the first k columns of U and $\tilde{U}$, that are eigenvectors of $M_2M_2{}^\top$ and $\tilde{M}_2\tilde{M}_2{}^\top$, obtaining $U_k = [u_1, .., u_k]$ and $\tilde{U}_k = [\tilde{u}_1, .., \tilde{u}_k]$. From Theorem B.1 we have that, for any $i = 1, ..., k$, holds

$$||u_i - \tilde{u}_i|| \leq 2^{\frac{3}{2}}\frac{2\epsilon||M_2||_F^2+\epsilon^2}{\min(\sigma_{i-1}(M_2)^2 - \sigma_i(M_2)^2, \sigma_i(M_2)^2 - \sigma_{i+1}(M_2)^2)} \leq 2^{\frac{3}{2}}\frac{2\epsilon||M_2||_F^2+\epsilon^2}{\alpha_{M_2}}$$

from which the thesis follows. □

The following results, taken from Yu et al. (2015) and Chen et al. (2012), present perturbation bounds on the eigenvectors and on the singular vectors of symmetric matrices.

Theorem B.1 (Cor. 1, pg. 4 Yu et al. 2015)**.** *Consider A and $\tilde{A}$ two symmetric matrices in $\mathbb{R}^{n\times n}$, with eigenvalues $\lambda_1 \geq ... \geq \lambda_n$ and $\tilde{\lambda}_1 \geq ... \geq \tilde{\lambda}_n$. Fix a $j \in \{1, ..., n\}$ and assume that $\min(\lambda_{j-1} - \lambda_j, \lambda_j - \lambda_{j+1}) > 0$, where we define $\lambda_0 = \infty$ and $\lambda_{n+1} = -\infty$. If $v \in R^n$ (resp. $\tilde{v}$) is an eigenvector of A (resp. $\tilde{A}$), associated to λ_i (resp. $\tilde{\lambda}_i$), then*

$$||v - \tilde{v}|| \leq \frac{2^{\frac{3}{2}}||A - \tilde{A}||}{\min(\lambda_{j-1} - \lambda_j, \lambda_j - \lambda_{j+1})}$$

Theorem B.2 (Thm. 3.2, Chen et al. 2012)**.** *Let $B \in \mathbb{R}^{k\times k}$ be a matrix, with SVD $B = Udiag((\sigma_1, ..., \sigma_k))V^\top$ with $\sigma_1 > \sigma_2 > ... > \sigma_k > 0$, and let $\tilde{B} = B + \Delta_B$ be a perturbed matrix, with SVD $\tilde{B} = \tilde{U}diag((\tilde{\sigma}_1, ..., \tilde{\sigma}_k))\tilde{V}^\top$.*
Define:

$$\alpha_B = \min_{i\neq j}|\sigma_i - \sigma_j| > 0, \quad \epsilon_B = \frac{\sqrt{2}||\Delta_B||_F}{\alpha_B}, \quad \gamma_B = \frac{||B||_F}{\alpha_B}(1 + \sqrt{1 - \frac{1}{k}})$$

Then, if

$$\epsilon_B \leq \frac{1}{2\gamma_B + \sqrt{1 + 4\gamma_B^2}} \tag{13}$$

The following upper bound holds:

$$||U - \tilde{U}||_F \leq \frac{\sqrt{2}\epsilon_B}{\sqrt{1 - 2\gamma_B\epsilon_B + \sqrt{1 - \epsilon_B^2 - 4\gamma_B\epsilon_B}}} \tag{14}$$

The following corollary is essentially a rewriting of the previous theorem.

Corollary B.1. *In the same setting of Theorem B.2, if there exists an $\epsilon > 0$ such that*

$$||\Delta_B|| < \epsilon \leq \frac{\alpha_B^2}{\sqrt{2}\left(2||B||_F(1 + \sqrt{1 - \frac{1}{k}}) + \sqrt{\alpha_B^2 + 4||B||_F^2(1 + \sqrt{1 - \frac{1}{k}})^2}\right)}$$

then $||U - \tilde{U}||_F \leq 2\sqrt{2}\frac{\epsilon}{\alpha_B}$.

Proof. Note that if ϵ satisfies the hypothesis of the corollary, then (13) is satisfied and hence we have (14):

$$||U - \tilde{U}||_F \leq \frac{\sqrt{2}\epsilon_B}{\sqrt{1 - 2\gamma_B\epsilon_B + \sqrt{1 - \epsilon_B^2 - 4\gamma_B\epsilon_B}}} \leq \frac{\sqrt{2}\epsilon_B}{\sqrt{1 - 2\gamma_B\epsilon_B}}$$

Now we plug in the last equation the bound of (13), to get

$$||U - \tilde{U}||_F \leq \frac{\sqrt{2}\epsilon_B}{\sqrt{1 - \frac{2\gamma_B}{2\gamma_B + \sqrt{1+4\gamma_B^2}}}} = \sqrt{2}\epsilon_B \sqrt{\frac{2\gamma_B + \sqrt{1 + 4\gamma_B^2}}{\sqrt{1 + 4\gamma_B^2}}} = \sqrt{2}\epsilon_B \sqrt{\frac{2\gamma_B}{\sqrt{1 + 4\gamma_B^2}} + 1} \leq 2\epsilon_B$$

□

References

Alighieri, D. (1979). *La Divina Commedia, a cura di N. Sapegno*. Nuova Italia, Firenze.

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. (2014). Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832.

Anandkumar, A., Hsu, D., and Kakade, S. M. (2012a). A method of moments for mixture models and Hidden Markov models. In *COLT*, volume 1, page 4.

Anandkumar, A., Liu, Y.-k., Hsu, D. J., Foster, D. P., and Kakade, S. M. (2012b). A spectral algorithm for Latent Dirichlet Allocation. In *NIPS*, pages 917–925.

Balle, B., Hamilton, W. L., and Pineau, J. (2014). Methods of moments for learning stochastic languages: Unified presentation and empirical comparison. In *ICML*, pages 1386–1394.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Chaganty, A. T. and Liang, P. (2013). Spectral experts for estimating mixtures of linear regressions. In *ICML*, pages 1040–1048.

Chen, X. S., Li, W., and Xu, W. W. (2012). Perturbation analysis of the eigenvector matrix and singular vector matrices. *Taiwanese Journal of Mathematics*, 16(1):pp–179.

Colombo, N. and Vlassis, N. (2016). Tensor decomposition via joint matrix schur decomposition. In *ICML*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pages 1–38.

Ge, R., Huang, Q., and Kakade, S. M. (2015). Learning mixtures of gaussians in high dimensions. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 761–770. ACM.

Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Hsu, D. and Kakade, S. M. (2013). Learning mixtures of spherical Gaussians: moment methods and spectral decompositions. In *ITCS*, pages 11–20. ACM.

Hsu, D., Kakade, S. M., and Zhang, T. (2012). A spectral algorithm for learning Hidden Markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480.

Huang, F., Perros, I., Chen, R., Sun, J., Anandkumar, A., et al. (2014). Scalable latent tree model and its application to health analytics. *arXiv preprint arXiv:1406.4566*.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218.

Jain, P., Jin, C., Kakade, S. M., Netrapalli, P., and Sidford, A. (2016). Matching matrix bernstein with little memory: Near-optimal finite sample guarantees for Oja's algorithm. *arXiv preprint arXiv:1602.06929*.

Jain, P. and Oh, S. (2014). Learning mixtures of discrete product distributions using spectral decompositions. In *COLT*, pages 824–856.

Kuleshov, V., Chaganty, A., and Liang, P. (2015). Tensor factorization via matrix factorization. In *AISTATS*, pages 507–516.

Liberty, E. (2013). Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM.

McDiarmid, C. (1989). On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188.

Newman, D., Asuncion, A., Smyth, P., and Welling, M. (2009). Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10(Aug):1801–1828.

Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110.

Song, L., Xing, E. P., and Parikh, A. P. (2011). A spectral algorithm for latent tree graphical models. In *ICML*, pages 1065–1072.

Stewart, G. (1990). Perturbation theory for the singular value decomposition. In *In SVD and Signal Processing, II: Algorithms, Analysis, and Applications*.

Stewart, G. and guang Sun, J. (1990). *Matrix Perturbation Theory*. Computer science and scientific computing. Academic Press.

Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Yu, Y., Wang, T., Samworth, R. J., et al. (2015). A useful variant of the davis–kahan theorem for statisticians. *Biometrika*, 102(2):315–323.

Zou, J. Y., Hsu, D. J., Parkes, D. C., and Adams, R. P. (2013). Contrastive learning using spectral methods. In *Advances in Neural Information Processing Systems*, pages 2238–2246.